ARYAMAN MISHRA

19BCE1027

1) #include<stdio.h>

#include<cstdlib>

struct node

{

int data;

struct node \*next;

struct node \*prev;

};

struct node \*head;

void create(int item)

{

struct node \*ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

if(head==NULL)

{

ptr->next = NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else

{

ptr->data=item;

ptr->prev=NULL;

ptr->next = head;

head->prev=ptr;

head=ptr;

}

printf("\nNode Inserted\n");

}

}

void search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

printf("\nEnter item which you want to search?\n");

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("\nitem found at location %d ",i+1);

flag=0;

break;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

if(flag==1)

{

printf("\nItem not found\n");

}

}

}

void printList(struct node\* node)

{

struct node\* last;

printf("\nPrinting the double linked list \n");

while (node != NULL) {

printf(" %d ", node->data);

last = node;

node = node->next;

}

}

int main ()

{

int choice,item,loc;

do

{

printf("\n1.Create\n2.Search\n3.Print the List\n4.Exit\nEnter your choice?");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the item\n");

scanf("%d",&item);

create(item);

break;

case 2:

search();

break;

case 3:

printList(head);

break;

case 4:

exit(0);

break;

default:

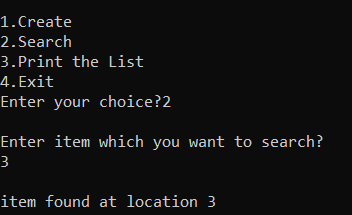
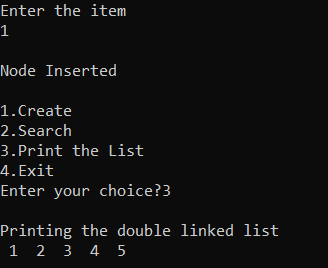
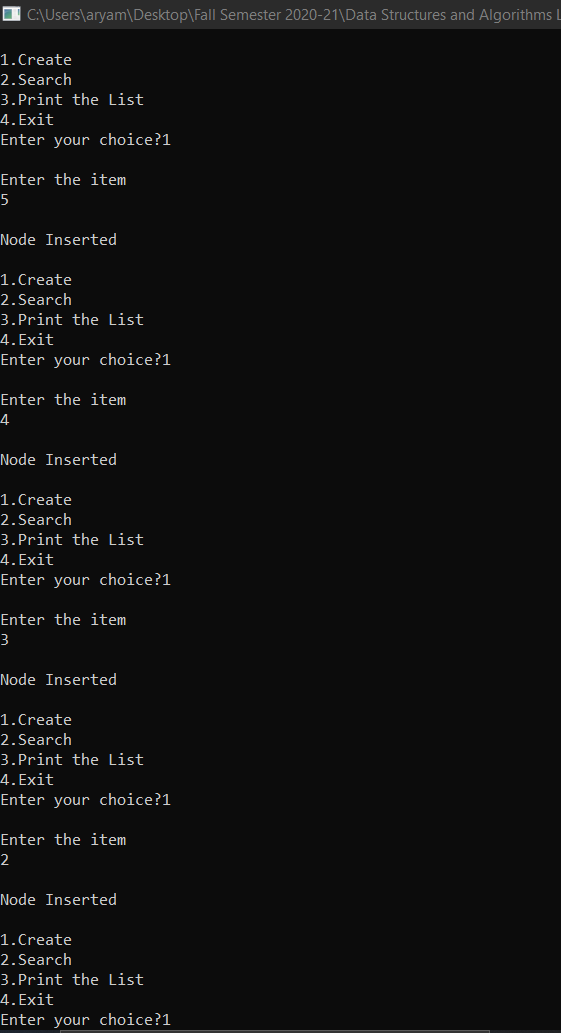
printf("\nPlease enter valid choice\n");

}

}while(choice != 3);

return 0;

}



2) #include <stdio.h>

#include<cstdlib>

//Represent a node of the doubly linked list

struct node{

int data;

struct node \*previous;

struct node \*next;

};

int size = 0;

//Represent the head and tail of the doubly linked list

struct node \*head, \*tail = NULL;

//addNode() will add a node to the list

void addNode(int data) {

//Create a new node

struct node \*newNode = (struct node\*)malloc(sizeof(struct node));

newNode->data = data;

//If list is empty

if(head == NULL) {

//Both head and tail will point to newNode

head = tail = newNode;

//head's previous will point to NULL

head->previous = NULL;

//tail's next will point to NULL, as it is the last node of the list

tail->next = NULL;

}

else {

//newNode will be added after tail such that tail's next will point to newNode

tail->next = newNode;

//newNode's previous will point to tail

newNode->previous = tail;

//newNode will become new tail

tail = newNode;

//As it is last node, tail's next will point to NULL

tail->next = NULL;

}

//Size will count the number of nodes present in the list

size++;

}

void update\_element(int data,int updated\_data)

{

struct node \* temp = (struct node \*)malloc(sizeof(struct node));

temp = head;

int pos = 0;

while(temp != NULL)

{

if(temp -> data == data)

{

temp -> data = updated\_data;

}

else

{

temp = temp ->next;

pos++;

}}}

//display() will print out the nodes of the list

void display() {

//Node current will point to head

struct node \*current = head;

if(head == NULL) {

printf("List is empty\n");

return;

}

while(current != NULL) {

//Prints each node by incrementing pointer.

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

int main()

{

int n,i;

//Add nodes to the list

addNode(10);

printf("Enter coins labelled 15, 20, 25, 30, 35 between 10 and 40 in any order while placing them in middle.\n");

for(i=1;i<=5;i++)

{

printf("Enter a coin.\n");

scanf("%d",&n);

addNode(n);

}

addNode(40);

printf("Old Chain:\n");

display();

update\_element(20,200);

update\_element(40,400);

printf("New Chain:\n");

display();

return 0;

}

